## AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions and listings of claims in the application:

### Listing of Claims

1. (Currently Amended) A code generator for generating an orthogonal code having a spreading factor (SF) and an index (k), wherein the spreading factor (SF) is selectable from values in a range $1 < SF \leq SF_{max}$ with $SF_{max}$ denoting a fixed maximum spreading factor, said code generator comprising:

an index conversion unit for converting the index (k) into a modified index (j) associated with a corresponding code having the fixed maximum spreading factor; and

a logic unit for solely performing logic operations on bits of the modified index (j) and bits of a counter value (i), thereby generating a code bit of the orthogonal code.

2. (Previously Presented) The code generator according to claim 1, wherein said corresponding code is one of: an orthogonal variable spreading factor (OVSF) code, a Hadamard code, and a Walsh code.

3. (Previously Presented) The code generator according to claim 1 wherein said index conversion unit includes multiplication means for multiplying the index (k) with a value of $SF_{max}/SF$.

4. (Previously Presented) The code generator according to claim 3, wherein said multiplication means includes:

a mapping unit for mapping the spreading factor (SF) to a number (s) equal to $\log_2\{SF_{max}/SF\}$,

a shift register adapted to receive and store the index (k) in binary representation, further adapted to receive the number (s) and to shift the stored index (k) by (s) bit positions in the direction of more significant bit positions.

5. (Previously Presented) The code generator according to claim 1, wherein the index conversion unit includes a permutation unit for permuting the bits of the index (k).

6. (Previously Presented) The code generator according to claim 3 wherein said index conversion unit includes:

a permutation unit for permuting the bits of the index (k); and

selection means for selecting, depending upon a mode signal indicating a desired type of said orthogonal code, the output of the permutation unit or the output of the shift register, thereby generating the modified index (j).

7. (Previously Presented) The code generator according to claim 1, wherein said logic unit includes:

adding means for performing binary AND operations, wherein the adding means is adapted to receive a bit of the modified index (j) and a bit of the counter value (i), and is further adapted to output a binary output value representing a binary AND combination of the two bits; and

combining means for combining the binary output values into the code bit.

8. (Previously Presented) The code generator according to claim 7, wherein said combining means includes means for performing binary XOR operations.

9. (Previously Presented) The code generator according to claim 1, further comprising a counter for generating the counter value (i).

10. (Currently Amended) A parallel code generator for concurrently generating a number $p > 1$ orthogonal codes having respective spreading factors $(SF_1, ..., SF_p)$ and indices $(k_1,..., k_p)$, wherein the spreading factors are selectable from values in a range $1 < SF_1, ..., SF_p \leq SF_{max}$, with $SF_{max}$ denoting a fixed maximum spreading factor, said parallel code generator comprising:

a number (p) of code generators, each for generating one of the p orthogonal codes having a particular one of the spreading factors and a particular one of the indices, each of said (p) code generators including:

an index conversion unit for converting the index (k) into a modified index (j) associated with a corresponding code having the fixed maximum spreading factor; and

a logic unit for solely performing logic operations on bits of the modified index (j) and bits of a counter value (i), thereby generating a code bit of the orthogonal code; and

a counter for generating the counter value (i) to be used by the (p) code generators.

11. (Currently Amended) A parallel code generator for concurrently generating a number $p > 1$ orthogonal codes having respective spreading factors $(SF_1, \ldots, SF_p)$ and indices $(k_1, \ldots, k_p)$, wherein the spreading factors are selectable from values in a range $1 < SF_1, \ldots, SF_p \leq SF_{max}$, with $SF_{max}$ denoting a fixed maximum spreading factor, said parallel code generator comprising:

a number (p) of code generators, each of said code generators including:

an index conversion unit for converting the index (k) into a modified index (j) associated with a corresponding code having the fixed maximum spreading factor;

a logic unit for solely performing logic operations on bits of the modified index (j) and bits of a counter value (i), thereby generating a code bit of the orthogonal code; and

a counter for generating the counter value (i);

wherein each of the code generators generates one of the (p) orthogonal codes having a particular one of the spreading factors and a particular one of the indices.

12. (Currently Amended) A method of generating an orthogonal code having a spreading factor (SF) and an index (k), wherein the spreading factor (SF) is selectable from values in a range $1 < SF \leq SF_{max}$, with $SF_{max}$ denoting a fixed maximum spreading factor, said method comprising the steps of:

a) converting the index (k) into a modified index (j) associated with a corresponding code having the fixed maximum spreading factor;

b) initializing a counter value (i);

c) solely performing logic operations by a logic unit on bits of the modified index (j) and bits of the counter value (i), thereby generating a code bit of the orthogonal code;

d) incrementing the counter value (i) by one ; and

e) repeating steps c) and d) until a desired number of code bits has been generated.

13. (Previously Presented) The method according to claim 12, wherein said corresponding code is one of: an orthogonal variable spreading factor (OVSF) code, a Hadamard code, and a Walsh code.

14. (Previously Presented) The method according to claim 12 wherein step a) includes multiplying the index (k) with a value of $SF_{max}/SF$.

15. (Previously Presented) The method according to claim 14 wherein said step of multiplying includes the steps of:

mapping the spreading factor (SF) to a number (s) equal to $\log_2\{SF_{max}/SF\}$;

storing the index (k) in binary representation in a shift register; and

shifting the stored index (k) by (s) bit positions in the direction of more significant bit positions.

16. (Previously Presented) The method according to claim 12, wherein step a) includes permuting the bits of the index (k).

17. (Previously Presented) The method according to claim 14 wherein step a) includes the steps of:

permuting the bits of the index (k); and selecting, depending upon a mode signal indicating a desired type of the orthogonal code, the permuted index or the shifted index, thereby generating the modified index (j).

18. (Previously Presented) The method according to claim 12, wherein step c) includes the steps of:

performing binary AND operations wherein each operation is adapted to combine a bit of the modified index (j) and a bit of the counter value (i), and to output a binary output value representing a binary AND combination of the two bits; and

combining the binary output values into the code bit.

19. (Previously Presented) The method according to claim 18, wherein said step of combining includes performing binary XOR operations.

20. (Canceled)

21. (Currently Amended) A computer program ~~product directly loadable into~~ loaded on an internal memory of a communication unit, ~~said product~~ comprising software code portions ~~that generate~~ for generating an orthogonal code having a spreading factor (SF) and an index (k), wherein the spreading factor (SF) is selectable from values in a range 1 < SF ≤ $SF_{max}$, with $SF_{max}$ denoting a fixed maximum spreading factor, ~~wherein,~~ wherein the software code portions perform the following steps when the ~~product~~ computer program is run on a processor of the communication unit~~, the following steps are performed~~:

a) converting the index (k) into a modified index (j) associated with a corresponding code having the fixed maximum spreading factor;

b) initializing a counter value (i);

c) solely performing logic operations on bits of the modified index (j) and bits of the counter value (i), thereby generating a code bit of the orthogonal code;

d) incrementing the counter value (i) by one; and

e) repeating steps c) and d) until a desired number of code bits has been generated.